

AFRL-IF-RS-TR-2002-316
Final Technical Report
January 2003



INVESTIGATING NETWORK INTRUSION

Booz-Allen and Hamilton, Incorporated

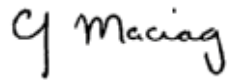
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2002-316 has been reviewed and is approved for publication.

APPROVED:



CHESTER J. MACIAG
Project Engineer

FOR THE DIRECTOR:



WARREN H. DEBANY, Technical Advisor
Information Grid Division
Information Directorate

REPORT DOCUMENTATION PAGE			<i>Form Approved</i> OMB No. 074-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE January 2003	3. REPORT TYPE AND DATES COVERED Final Aug 98 – Feb 99	
4. TITLE AND SUBTITLE INVESTIGATING NETWORK INTRUSION			5. FUNDING NUMBERS C - F30602-98-C-0253 PE - 33140F PR - 7820 TA - 05 WU - 02	
6. AUTHOR(S) Julia Pilny				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Booz-Allen and Hamilton, Incorporated 8283 Greensboro Drive McLean Virginia 22102-3838			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory/IFGB 525 Brooks Road Rome New York 13441-4505			10. SPONSORING / MONITORING AGENCY REPORT NUMBER AFRL-IF-RS-TR-2002-316	
11. SUPPLEMENTARY NOTES AFRL Project Engineer: Chester J. Maciag/IFGB/(315) 330-3184/ Craig.Maciag@rl.af.mil				
12a. DISTRIBUTION / AVAILABILITY STATEMENT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.				12b. DISTRIBUTION CODE
13. ABSTRACT (Maximum 200 Words) In today's information-hungry world, vast amounts of data pass through AF networks on a daily basis. Information Assurance, and the Air Force Enterprise Defense (AFED) product in particular, concerns itself with the protection of these networks and associated data. It is not unusual for security databases on large networks to add a million new records per day. The specific size and nature of these data is dynamic, depending greatly upon the number of network security sensors and the network load. While the static administrative data does not impose abnormal loads on the database, the more dynamic network data imposes stress on the database due to its unpredictable volume size and insertion rates. When the database contains millions of records, performance can suffer. To keep the database from becoming full or fragmented, these data records must be offloaded at certain intervals. Archiving and then removing network data causes database table sizes to fluctuate, which in turn can impact performance because indexes must be recalculated. Clearly, it is an imposing task to keep the database performing at its optimum. The goal of this paper is to discuss current storage and access methods, their advantages and shortcomings, what new database technologies are available, and what direction the database development should take to best serve the IO/IA environment.				
14. SUBJECT TERMS Database, DBMS, Information Assurance, Intrusion Detection, Archival, Partitioning, JEFX, AFED, EPIC, Cartridges, Syntax, Semantics, Data Mining, Statistics, Visualization			15. NUMBER OF PAGES 17	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL	

Table of Contents

SECTION 1. AFED and Traditional Database Implementation	1
1.1 Introduction	1
1.2 Current AFED Database	2
SECTION 2. Current Performance Solutions	3
2.1 Storage and Retrieval.....	3
2.2 Partitioning	4
2.3 Cartridges	6
SECTION 3. Future Techniques and Goals.....	7
3.1 Syntax and semantics limit interaction/knowledge.....	7
3.2 Statistical Behavior.....	9
3.3 Data Mining.....	9
3.4 Additions to Data Mining	11
SECTION 4. Conclusion	13

List of Figures

Figure 1 Solutions	5
Figure 2 – Mapping Activity from Source IP to several Destinations IPs.	12

SECTION 1. AFED and Traditional Database Implementation

1.1 Introduction

In today's information-hungry world, vast amounts of data pass through various networks on a daily basis. Information Assurance, and the Air Force Enterprise Defense (AFED) product in particular, concerns itself with the protection of these networks and associated data. In the course of accomplishing such a task, IA efforts themselves can produce large amounts of information as a by-product of collecting Transmission Control Protocol (TCP) and Universal Datagram Protocol (UDP) data packets for analysis and subsequent storage. The specific size and nature of these data is dynamic, depending greatly upon the number of network security sensors and the network load. It is not unusual for security databases on large networks to add a million new records per day.

The current version of the AFED database provides a multi-purpose data store for analysts and administrators who can not only access information on network activity, but also host-based change management information and other static data (relating to users, missions, services, vulnerabilities, and policy violations).

While the static administrative data does not impose abnormal loads on the database, the more dynamic network data imposes stress on the database due to its unpredictable volume size and insertion rates. When the database contains millions of records, performance can suffer. To keep the database from becoming full or fragmented, these data records must be offloaded at certain intervals. Archiving and then removing network data causes database table sizes to fluctuate, which in turn can impact performance because indexes must be recalculated. Clearly, it is an imposing task to keep the database performing at its optimum.

The goal of this paper is to discuss current storage and access methods, their advantages and shortcomings, what new database technologies are available, and what direction the database development should take to best serve the IO/IA environment.

1.2 Current AFED Database

The AFED database deals with two distinctly different types of data; somewhat predictable static management data, and dynamic real-time network data.

The static data (sensor signatures, normalized signatures, sites, known services, change management information such as hostnames and the services available on each host) is loaded through various means. Scripts are established to load flat files containing sensor signatures, normalized signatures, etc., while a Practical Extraction and Report Language (PERL) script is used to populate all change management information.

The dynamic data is provided to the database from sensors that pass formatted data via software bridges. Any number or type of sensors can be utilized provided the data they supply are formatted correctly.

As the data is being ingested, database triggers perform a lookup of what each sensor is reporting to correlate those into a single event. The more sensors that are incorporated, the greater the processing load. Performance is affected by writes to the session table, event table, and log files, reads from the signature tables, and trigger processing, all contending for system resources.

A greater amount of dynamic data may be supplied in the future from additional sources. A branch within the Information Directorate, IFGA, has a project that detects suspicious activity from a network hardware perspective. An interface is being developed that will import suspicious activity detected from routers to the database. Under The Technical Cooperation Program (TTCP), there is an initiative to share data with 2nd party Intrusion Detection Systems (IDS). In time, there may be a direct feed from those systems, supplying alarms to the AFED database. Another possible dynamic input is from protocols other than TCP/IP and UDP, such as SNMP.

As each of these inputs becomes incorporated into AFED, the volume of data will greatly increase, raising the performance issues as witnessed during the Joint Expeditionary Forces eXercises (JEFX) in September of 1998 and 1999. During those exercises, the average number of records written to the database averaged 400,000 per day. This resulted in an overloading of system resources and populating database tables beyond efficient levels, thus making queries more difficult. It also presented database administrators with a familiar problem: how to best offload the data, and how often.

Achieving a lightweight database means archiving and deleting data more frequently. At the first JEFX, data was offloaded on a daily basis. This exposed two inadequacies, as outlined below.

Once the data was exported, the active tables were truncated to free storage space for the next day's exercise. The previous day's exported data was then off-line and not available for review or for final analysis.

Secondly, the export/import processing of data is slow and fraught with pitfalls. The preserved data must be imported to the database again in order to access it, which exposes more problems due to non-unique keys. (The table that stores the sensor information has a unique sensor session ID key. Sensors were cycled when the data was exported, so when the sensor was restarted, the Session ID was set back to 1. Upon importing, the exported data keys were now duplicates of those in the active data, causing errors. To circumvent the problem, triggers were used to alter the session ID on insert, but the Session ID was a foreign key to other tables, increasing the level of complexity.)

For the second JEFX exercise in 1999, a Trend Analysis database was provided in which older data was "boiled down" or condensed, and transferred onto a database on a separate drive.

Building the Trend Analysis database was a very processing-intensive activity. It involved finding all records belonging to the same session and writing one record summarizing the activities. A PL/SQL routine was written that looked for session records within the archival date and time, then read all other records that shared the same session ID. Each activity had to be analyzed and consolidated, the trend record written, and finally the session records deleted. Complex problems arose, including what to do with a session that has no START, or more so, what to do with a session having no CLOSE activity. If there was no START, the session was considered to have been open upon startup of the system, and that session information was dropped since the trend record would be incomplete. Records that had no CLOSE were considered to still be open, and those records were left in the AFED database. But if that session were actually no longer active, the routine would repeatedly find those records and never delete them.

Many of these issues were not resolved at JEFX. However, since the Trend Analysis routine is so time consuming, other approaches are being considered. Section 2 discusses other considerations, as well as goals affiliated with the AFED database.

SECTION 2. Current Performance Solutions

2.1 Storage and Retrieval

Before delving into new ways to view data, new techniques for storage and retrieval that eliminate some of the performance issues should be discussed. In Section 1, performance under the current architecture was demonstrated to be a conspicuous problem. Several options exist that may alleviate this. One is through the use of *partitioning*, and the other is through use of a time series *cartridge*. Both of these techniques provide streamlined or faster access to data, meaning better performance.

One concept that has not been addressed, but does answer some of the performance problems, is to incorporate more horsepower, running both parallel and replication servers. The idea of

adding more computers and disks is unfortunately a luxury that most budgets cannot support at this time.

2.2 Partitioning

Partitions are designated segments of storage in a file system. They add further granularity by pinpointing the size, location, and number of segments or partitions to create. When a database is created, storage is set up using tablespaces that allocate a specified amount of disk space for data in a specified location. Each partition can be assigned to a different tablespace, so a single table can be created across multiple partitions, each residing on its own tablespace. As a database becomes very large, the administrative tasks of managing the data become extremely challenging, as alluded to earlier. For example, it is much easier to handle four 25GB parts of a table than one 100GB table. Dividing a table into multiple partitions has many benefits, such as easier recovery from failure, load balancing by placing partitions across different disks, and overall performance. Partitions can be taken on and off-line, swapped with other partitions, and new partitions can be added, all the while having the rest of the table accessible.

There are three partitioning mechanisms provided by Oracle 8i: hashing, composite, and range. The optimal mechanism for the AFED application is to use partitioning by *range*, since this technique allows data to be separated by date. The other two options (hashing and composite) are outside the scope of this paper.

Range partitioning is useful for data warehousing efforts that perform maintenance on large volumes of data, and applications that maintain a scrolling window of data. It also lends itself to operational work on subsets of data. Within AFED, the analyst will typically be working on the most recent sensor and event data, while the older data has been dispositioned and then off-loaded, making this technique the best-fit scenario. The primary benefit to the AFED program is that a partition can be taken off-line, while the rest of the table remains on-line. Capitalizing on performance, the analyst can be working with the current data, the sensors can be feeding data in, while the oldest partition is being ported to a trend analysis database and removed from the AFED database, helping to keep it as lightweight as possible.

As the name implies, range partitioning makes use of ranges to set the boundaries for each partition. Each data row is written to the partition where the attribute value falls inside the partition's range. Any date attribute can be used, so event and sensor data can be written into partitions based on the date the record was created. Indexes are also partitioned in the same manner as the partitioning algorithm.

There are rules for partitioning that make design for the AFED task unintuitive. This is due to the design of range partitions where:

- Every partition of a range-partitioned table or index has an exclusive upper bound, which is specified by the VALUES LESS THAN clause.

And

- Every partition except the first partition also has an inclusive lower bound, which is specified by the VALUES LESS THAN on the next-lower partition.

This means "GREATER THAN", "BETWEEN", and "EQUAL" are not accepted operators to set up the partition boundaries. A partition cannot be created where the creation date is *greater than* yesterday's date. This is due to the nature of data warehouses and the flow of data. Partitioning tends to be implemented *at the data warehouse* where an on-line transaction database has first manipulated and indexed the data before adding it into the partition pool. Conversely, AFED should take advantage of the partitions as a solution for performance problems associated with maintenance and large volumes of data *at the on-line* database, as shown in Figure-1. This does not mean that range partitions may not be useful, but that investigating the proper setup will be more complicated. Also, the column used to sort the data into the partitions cannot be manipulated. Therefore, a function like `to_date(column,format)` cannot be used on the range column. If it is desired to partition the data by hours or days, that information cannot be extracted using the `to_date` function to return a day or hour format.

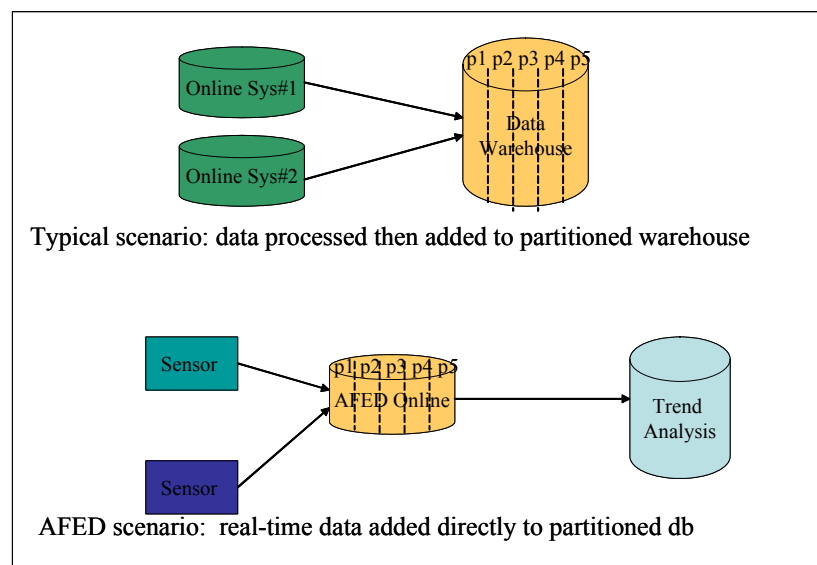


Figure 1 Solutions

In the AFED example provided below, a column is added to the table named DB_HR that contains the hour the data was created, since the hour cannot be extracted from the creation date. The partitions are set up to capture every six hours of data, or to quarter up the day.

```
CREATE TABLE SESSIONS (SESSION_ID, CREATE_DT, DB_HR.....)
PARTITION BY RANGE (DB_HR)
(PARTITION Q1
VALUES LESS THAN (7) TABLESPACE T1
PARTITION Q2
VALUES LESS THAN (13) TABLESPACE T1
PARTITION Q3
VALUES LESS THAN (19) TABLESPACE T1
PARTITION Q4
VALUES LESS THAN (25) TABLESPACE T1)
```

This would allow maintenance to occur on a single partition, while the other partitions are still available. There are two consequences to this approach. One is that partitions are like walls, and querying the full table causes each partition, in a practical sense, to be "joined". This may be worthwhile, as the application can be tailored to read the most recent partition, and it is projected that the analyst will seldom scan the entire table. The analyst is primarily investigating current activities and typically only accessing older data to substantiate or eliminate theoretical suspicious behavior. Therefore, the cost hit of joining the whole table across the partitions should rarely come into play, and may be a viable means of maintaining the data. The other consideration is that while the oldest partition is undergoing maintenance, that data is unavailable to the analyst. Once the data has been condensed, it will be placed into the trend analysis database, making it available. However, if that information is crucial in analyzing an attack, no matter how short a time the maintenance may take, it still may not be acceptable to have unattainable data. A decision must be made at this point whether the trend analysis database will be a collection of off-line data stores to be imported at a later date into an on-line system, or a separate disk drive (or a second machine).

2.3 Cartridges

Cartridges are another technology that shows potential for the AFED program. Cartridges are essentially library routines that provide utilities for specific applications. Historically, cartridges were side products that were purchased separately, but today the majority of the cartridges have been incorporated into the Oracle 8i kernel. There are many types of cartridges, dealing with time, space, text, etc. The time series cartridge may present AFED with another option for handling large volumes of data being processed in a timely manner. Specifically, the time series is an extension to Oracle8i that provides storage and retrieval of timestamped data through object types. With Oracle8i Time Series, data can be managed more conveniently and efficiently than is possible using traditional data types and user-defined functions. Oracle8i Time Series can store time series data in the database under transactional control. Once stored in the database, this data can be retrieved by finding a row in the dynamic table that contains the primary key, which must include the timestamp.

This parallels the partitioning aspect mentioned above because the primary attribute driving the separation of the lightweight database and the trend analysis database is time. Putting the timestamp as the key and utilizing the library functions developed within the Time Series cartridge allows faster retrieval for maintenance. This technique is also useful for a single computer environment. If there were no replication or parallel server options turned on, then maintenance routines would have to archive data to some backup medium and then delete the archived records from the current database. With timestamp keys and utilities, it is much more efficient to specifically retrieve, archive and delete records within a certain time period.

The disadvantage to using Time Series is that the tables that are dynamic are already heavily indexed. Relationships are required to many other tables; indexes have been placed on fields to support foreign keys for these relationships. The graphical user interface (GUI) for the analyst also provides a filtering capability, which requires indexes for performance purposes. The main

methods of sorting the data are very different from the unique key and foreign key derivations. Placing the timestamp as part of the primary key and indexing on the timestamp creates the possibility of six indexes on one table. This in itself is a performance impact, especially with a high rate of inserts and queries. The more indexes that exist on a table, the lower the performance quality, as each index must be sorted for a new entry or update. However, compared to the original method of performing maintenance row by row on voluminous data, this approach has merit and deserves consideration.

Whether the AFED solution is partitioning, using the Time Series cartridge or a combination of both, remains to be seen. However, applying new database technologies to address real-time large database concerns will resolve some of the current database issues. Section 3 discusses situations in which the database can provide more than mere storage. Some new concepts are presented, along with what possible database techniques may be applicable to demonstrate these concepts. The following concepts and techniques are offered with the premise that the AFED database functions are within respectable performance requirements, and the data is now placed into a large trend analysis database.

SECTION 3. Future Techniques and Goals

Not immediately obvious is the wealth of information within the culminated database that comes from higher levels of abstraction. Some of these new techniques involve looking at large amounts of data from different perspectives, breaking the barriers that come with tradition, to see how data clusters together outside of the confines of designed relationships. These future techniques and how they can increase AFED's usefulness to the analyst are discussed below.

3.1 Syntax and semantics limit interaction/knowledge

How much data does the human mind absorb by 30 years of age? How much information other than obvious (already abstracted) intrusion information exists in an AFED trend analysis database? The abstract usage's and current understanding of databases parallel in many ways the human brain. If one thinks of this large accumulation of Intrusion Detection information as a catalog of a person's mind, then some extremely valuable knowledge can be formulated. However, there are some obstacles to true understanding of this data. One is syntax. With a large database, there is no way to easily correlate and retrieve information that heuristically leads to an intrusion.

Currently, the syntax problem centers on the fact that there is no way to easily retrieve on-going trends and also non-trends. The following are examples of each, starting with trends. A denial of service is a technique where a computer gets flooded by requests for services. Although detected by a sensor/sensors, who activated or how long ago a previous denial of service occurred is data that may be recorded, but not easily retrieved, nor correlated and presented to the analyst. The next denial of service has the same relevance and priority as the last, and it is up

to the analyst(s) to determine if this is recurring or not. If it is recurring over a one month span, from the same originating hosts to a different target host, are analysts going to detect this behavior? The database has this information, but how can that buried information be abstracted? This is an example of an on-going trend because the analysts will be alerted to what activities are paramount, but no correlation is happening to point out similar past behaviors.

As an example of non-trended information, consider the following. Sensors do not report a single PING from an outside network to a host within our network as an intrusive event. Many single PINGs may occur from the same outside source spaced over a long period of time that are never flagged. Over a week's time, dozens of hosts may have been identified within the same network, and not a single sensor has detected this. It is clearly feasible that the entire network is being mapped by calculated activities like this that are seemingly harmless. However, some of the sensors do record this information even though the analyst is not presented with an alarm. So, potentially, all that information can be retained in the trend analysis database, since packet level information (dependent on which sensors are employed) is recorded.

Comprehending all reported data produces knowledge. If this type of situation can be identified, then the outlying source(s) that PINGed the internal hosts can be identified as bad IP addresses and flagged. Any new activity on the network originating from that bad IP can be immediately placed as an intrusive event. Counter measures can then follow to thwart these pursuits or sever connections from those hosts.

As the importance of discovering what knowledge is in the data becomes apparent, underlying questions relevant to the database start to formulate. How do we get to this information? It is now understood that the data holds enormous amounts of information waiting for human extraction. The problem is two-fold. What are the questions that should be asked, and how do we correctly direct those questions over a very extensive amount of data?

Consider the PING example. The first action might be to retrieve all PINGs to hosts on our network within a month. This is incorrect, because the PINGs should be from the same source IP. Or do they need to be from the same source? Attackers can be using other compromised hosts to send PINGs so even the last hypothesis may not work. However, there is something to be obtained from an inordinate amount of PINGs occurring on a host within a single network whether from a single source or not. So now the "trend" of behavior brings relevance to intrusive or normal behavior.

The typical database paradigm confines day to day usage to questions that are translated into selection statements in standard query language (SQL), or by further processing data using some call interface with a more robust programming language interface. Formulating and translating correctly constructed semantics and incorporating the proper syntax to get correct information from a database is a difficult process. Unfortunately, this is a normal activity of database processing. It parallels everything from speech recognition to the writing recognition alphabet used in Palm Pilots. The user has to be trained to "speak" the language of the tool. The same is true of databases. SQL is the consistent method for database information retrieval. However, in the situations where there exist billions of records, standard SQL will not meet the performance needs of the end user.

If normal database disciplines of selection can be shifted to current trends, and how to abstract these trends, then the questions to be asked are more readily answered. Trends can be abstracted through statistical methods or depicted through visualization techniques. Some of the various possibilities to discover trends are discussed in section 3.2 of this paper.

3.2 Statistical Behavior

Statistics is the collection, organization, and interpretation of numerical data, especially the analysis of population characteristics by inference from sampling. Typical usage for statistics involves probabilities, cluster algorithms, bell curves, etc., that assign a certain weight to a known data point. The underlying characteristic of statistics reports information on how data relates to the other data points in the sample. Compare what relational database traits are to what traits statistical methods inherently use. Relational databases store data together that have "like" characteristics, or that have a relationship. The underlying premise with a database is that the payoff comes when data is stored and retrieved. Since all data in a table or in related tables are logically linked to each other, then disk storage and retrieval is most likely condensed, since related objects reside in the same table, and will be written or retrieved together. This then reduces I/O and CPU resources by storing data in one set of sequential storage blocks rather than peppering data all over the total disk storage.

The differing statistical methods are similar on a larger scale. Methods have been developed that are used to associate uncertainties with inferences drawn from data. These methods help to describe relationships within the data points. At a high level, since the underlying reason AFED is using a relational database is to store this data, then it may be concluded that further relationships from statistical methods may uncover the more covert activities.

3.3 Data Mining

Data Mining is a relatively new field where the emphasis is not on the collection and organization of the data, but on interpreting those collections and turning that into information. There are many data mining tools in existence today. Data mining tools retrospectively look at data samples and apply statistical methodologies against them. They are very powerful tools. Uncovering hidden patterns, trends and relationships inside noisy data is an extremely important step in strengthening Information Assurance. However, there are some drawbacks to Data Mining for AFED that must be understood.

Most commercial off the shelf (COTs) Data Mining tools have some restrictions that poorly lend themselves to the AFED project. AFED has a large textual database, in which everything from the source and destination IP addresses to the signature of the attack is in a character based format. Most Data Mining tools rely on a numerical representation of data. This means that for AFED, every textual field would have to be represented by a "mapped" number.

Another restriction of most data mining applications is the limitation over the number of representations for data. Most commercial data mining applications are focused on a small range of options depicting the data, and used for depicting trends over large amounts of data with few variables. For example, most data mining tools are geared towards internet based electronic-commerce applications. These applications utilize pages written in hypertext mark up language (HTML), and minimize the amount of data inputted to a database. More specifically, these pages incorporate many pull-down menus that define a finite set of values. As an example, the most widely varied option would be the State of the resident, so the largest mapping is location to buying trends with 50 possible values. Most tools focus on small ranges for data values due to the nature of statistical analyses. The larger the amount of data points, the harder it is to find patterns and relationships amongst the data. Because of this, most tools have an upper limit to the number of mappings (typically between 200 and 300 unique mappings).

AFED incorporates a technique that identifies each suspicious activity reported by every sensor as one type of attack, or the "signature" of the attack. There are currently approximately 300 unique attack signatures. As new operating systems, hardware platforms, and the number of computer users increase, the number of ways to attack machines will increase. In the data mining environment, each attack signature has to be mapped to a number. Obviously, this does not lend itself well to the AFED data set. If the number of signatures is greater than the allowed number of mappings, this primary indication of intrusive activity can not be mined.

General COTS data mining techniques are not real-time and do not hook into a database. The means for mining a database is to select a sample from the database, convert the textual fields to numeric values, and place the resultant sample in a flat file. Only then can the data mining algorithms work. The application must also know what the report columns, input columns, and output column(s) are. For AFED to incorporate the typical data mining tool implies a tremendous amount of work, not only with the learning curve to correctly use the tool, but also the overhead of preparing the data. The difficulties will be because of the restrictions in mapping textual data to numeric representations, and because sampling and mapping a two billion record database into even 500,000 records for mining is not a trivial task.

How is a statistical sample determined to be a good representation of the original data? The sample has to include enough attack and non-aggressive behavior to characterize the possible activities occurring on the network. Is it possible to get an even distribution that characterizes each known attack in a sample database? If each signature is forced for inclusion, then there may be a skewed weight distribution or probability of an occurrence, since all occurrences were forced to exist in the sample database. If the sample is taken randomly, then how can the sample truly represent the attack possibilities? Consider, for example, the fact that every human must have brown, blue, green, or hazel colored eyes. If in all the records used to mine data related to eye color, there was not a single record of blue eyes to build a sample, then how good is the sample? How can someone with blue eyes be recognized in the context of the sample data? If AFED has 300 possible attack signatures and only 100 get characterized in the sample database, how will the sample explain unsampled attack activities?

There are many statistical algorithms that are used in data mining. Algorithms vary from simple clustering techniques to neural networks, linear regressions, logistic regressions, etc. Another

learning curve parameter is determining which is the best-fitting statistical method for AFED data.

The last concern with current data mining techniques is that the data sample will become stagnant. In ever evolving computer technology, we can expect new Trojan horses, viruses, denial of services, and compromises to be forthcoming. Technology is its own Pandora's box, in that the more it progresses, the complex associated problems become more progressive. Having a static model represent activities in an ever changing technological world is unrealistic. The question becomes how practical is it to continually re-sample, convert/map the data to numeric representation, build a flat file, and test the effectiveness of the sample? How often should it be done?

Although there is much overhead and a large learning curve associated with using data mining techniques, there are significant benefits. The first is that once a sample model has been produced, it can be made into a run time model (for most data mining applications). Current data, mapped numerically the same as the model, can be projected against the run time model. This allows the analyst the opportunity to have suspect activity posed against a statistical model to help evaluate the weight of the attack.

Another benefit is that the resultant information can be stored into the current database. This allows a sort of recursive learning to occur. If a process were in place to refresh the statistical sample on some time interval, run batches of new data against the model, and store the statistical measures in the database, then there is a tremendous potential for learning from the data. Looking at the analysts' dispositions, along with probabilistic results, can allow us to differentiate the false alarms from truly serious threats.

3.4 Additions to Data Mining

There are some data mining techniques that break the COTs "best statistical mining tool", or the normal conventions of data mining tools discussed in Section 3.3. Some applications can bring extra value to the AFED program by eliminating the real shortcomings of data mining as applicable with AFED. These problem areas are:

1. Lack of visualization,
2. Lack of database connectivity,
3. All algorithms are internal algorithms against a single model.

Some evolving data mining techniques address these prevalent problems. Although investigation thus far has not uncovered a data mining algorithm that directly pulls data from a connection, the other key problems have been addressed, as discussed below.

Most data mining tools return output value(s) in a report format. As mentioned above, this is useful when infused with the database data. But a report is not intuitive to the end user. There are some data mining applications that incorporate visualization techniques to display the multivariate structures. Below is a visual technique with some possible scenarios from a data mining technique developed by Booz Allen and Hamilton incorporating visualization tools. The

figure displays a scenario when a source is predominantly associated with a destination target. The blue source line indicates a specific source IP. The green lines show activity at specific destination IP addresses. The band of green lines indicates a large amount of activity associated with a few destination addresses. Such visualizations are useful for a number of reasons:

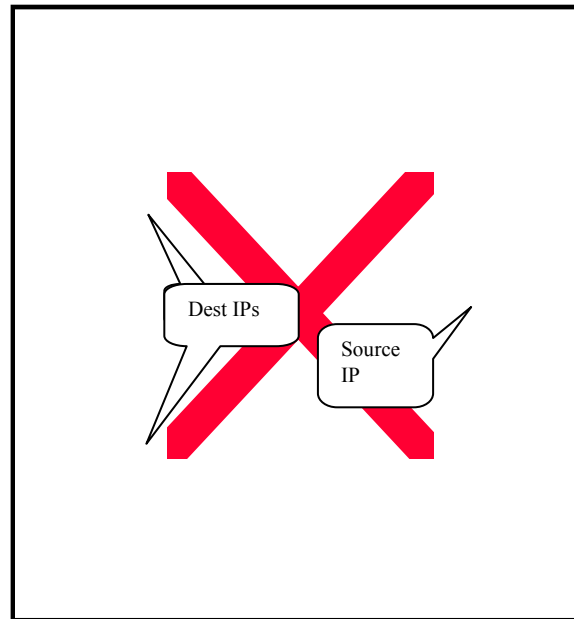


Figure 2 – Mapping Activity from Source IP to several Destinations IPs.

1. While the skew in traffic may be due to the fact that the destination machine hosts the organization's web server, customizing the visualizations for the web server to display events that are not associated with normal behavior can immediately show if the server is compromised.
2. The pattern may also indicate vulnerabilities associated with the particular source and trigger a response in the mind of the system administrator to look for the installations of latest patches. The server may be advertising itself as an "SMTP Mail Server –All" during a DNS lookup, giving an attacker unnecessary information about itself and hence, its weaknesses.

The results developed from the classification models will help to develop new rules that are very specific to AFED.

Most tools mine data points within the built data model. However, some use classification and clustering algorithms to gain insight into attack patterns, and then compare a normal network structure with the anomalous network structure. The advantage of this type of technique is that a suitable "peace time" model can remain fairly stagnant, and then a comparison can be made with current real network traffic.

SECTION 4. Conclusion

There are ever changing tools and techniques for retrieving data as we enter the 21st century. The original intent of this paper was to discuss all major tools, but the scope was narrowed to what tools are deemed "the most likely to succeed".

Further exploration may include ways to look into time and spatial relations in the data, perhaps using other Oracle cartridges. There may also be a way to map data to other visualization techniques. As an example of further visualization, consider if geographical coordinates of IP addresses could be visualized on a map. Then certainly an international connection originating from an adversarial country would be easily identified. Other techniques may require redefining "visual techniques" to include other sensory techniques. I have often entertained the idea of mapping data to sounds, applying innocuous patterns to the pitches just inside the human ear's audible range, with attack patterns in the center of the range. As a conductor hones in on an incorrect note amidst a hundred musicians playing, so could an analyst in time become aware of unfamiliar sounds. Sounds could be sped up or slowed down to discover intervals of patterns. An analyst could learn to intuitively know the comfort of white noise, and the threat of pulses or a steady pitch.

The ultimate goal is the ability to abstract all information required from a database with minimal effort, and display or arrange this data in a more intuitive manner. At the present, it is possible to retrieve this information through the use of some of the methods presented in this discussion, however, these endeavors are not trivial. At least with some of these techniques it is possible to achieve levels of abstraction and further assist the analyst with knowledge of what has and is happening through data collected.